Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **JUN 2015**	2. REPORT TYPE	3. DATES COVERED **00-00-2015 to 00-00-2015**

4. TITLE AND SUBTITLE
Matrix-Free Polynomial-Based Nonlinear Least Squares Optimized Preconditioning and its Application to Discontinuous Galerkin Discretizations of the Euler Equations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School,Department of Applied Mathematics,Monterey,CA,93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We introduce a preconditioner that can be both constructed and applied using only the ability to apply the underlying operator. Such a preconditioner can be very attractive in scenarios where one has a highly efficient parallel code for applying the operator. Our method constructs a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm. We show that this polynomial-based NLLS-optimized (PBNO) preconditioner significantly improves the performance of a discontinuous Galerkin (DG) compressible Euler equation model when run in an implicit-explicit time integration mode. The PBNO preconditioner achieves significant reduction in GMRES iteration counts and model wall-clock time, and significantly outperforms several existing types of generalized (linear) least squares (GLS) polynomial preconditioners. Comparisons of the ability of the PBNO preconditioner to improve DG model performance when employing the Stabilized Biconjugate Gradient algorithm (BICGS) and the basic Richardson (RICH) iteration are also included. In particular, we show that higher order PBNO preconditioning of the Richardson iteration (run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment. Because the NLLS-based algorithm used to construct the PBNO preconditioner can handle both positive definite and complex spectra without any need for algorithm modification we suggest that the PBNO preconditioner is, for certain types of problems, an attractive alternative to existing polynomial preconditioners based on linear least-squares methods.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**25**	

Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Matrix-Free Polynomial-Based Nonlinear Least Squares Optimized Preconditioning and its Application to Discontinuous Galerkin Discretizations of the Euler Equations

L.E. Carr III · C.F. Borges · F.X. Giraldo

February 12, 2015

Abstract We introduce a preconditioner that can be both constructed and applied using only the ability to apply the underlying operator. Such a preconditioner can be very attractive in scenarios where one has a highly efficient parallel code for applying the operator. Our method constructs a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm. We show that this polynomial-based NLLS-optimized (PBNO) preconditioner significantly improves the performance of a discontinuous Galerkin (DG) compressible Euler equation model when run in an implicit-explicit time integration mode. The PBNO preconditioner achieves significant reduction in GMRES iteration counts and model wall-clock time, and significantly outperforms several existing types of generalized (linear) least squares (GLS) polynomial preconditioners. Comparisons of the ability of the PBNO preconditioner to improve DG model performance when employing the Stabilized Biconjugate Gradient algorithm (BICGS) and the basic Richardson (RICH) iteration are also included. In particular, we show that higher order PBNO preconditioning of the Richardson iteration (run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment. Because the NLLS-based algorithm used to construct the PBNO preconditioner can handle both positive definite and complex spectra without any need for algorithm modification, we suggest that the PBNO preconditioner is, for certain types of problems, an attractive alternative to existing polynomial preconditioners based on linear least-squares methods.

Keywords preconditioning · polynomial preconditioner · nonlinear least squares · spectral elements · Galerkin methods · Euler Equations · nonhydrostatic atmospheric model

L.E. Carr III
Department of Applied Mathematics
Naval Postgraduate School
Monterey, CA 93943, USA
E-mail: lecarr@nps.edu

C.F. Borges
Department of Applied Mathematics
Naval Postgraduate School
Monterey, CA 93943, USA
E-mail: borges@nps.edu

F.X. Giraldo
Department of Applied Mathematics
Naval Postgraduate School
Monterey, CA 93943, USA
E-mail: fxgirald@nps.edu

Mathematics Subject Classification (2000) 65M60 · 65M70 · 35L65 · 86A10

1 Introduction

1.1 Background

In a variety of practical applications one must repeatedly solve a large system of linear equations where one has an extremely fast parallel code for applying an underlying fixed linear operator. Coincident with this, there are many iterative methods (e.g. GMRES) that can leverage this ability as they rely only on repeated applications of the operator and hence their speed is directly related to the speed with which the operator can be applied. Unfortunately, many of these methods converge very slowly without proper preconditioning which can be problematic. Although common approaches to preconditioning such as incomplete LU can be very effective in reducing the number of iterations, they are not, generally, able to exploit the same parallel structure as that of the original operator. This makes their use very troublesome because they may only be practical if one can construct a sufficiently fast code for applying them. Such a code may not be feasible for a variety of reasons (cost of development, parallel structure that is different from that of the original operator, etc.).

Fortunately, polynomial preconditioners do not suffer from these drawbacks as they can directly leverage the existing parallel code for the operator and hence are excellent candidates for problems of this type. With this in mind, our goal is to develop a method for constructing an effective polynomial preconditioner that requires *only* the existing code for applying the original operator, from construction to application. We will derive and then demonstrate the effectiveness of this approach on a specific test problem involving a rising thermal bubble (RTB). We considered this problem previously in [3] using an implicit-explicit (IMEX) 2-D continuous Galerkin (CG) model of the Euler equations in Schur-complement form with a *continuous* initial temperature distribution. Here we revisit the problem using a 2-D DG model (without Schur-complement form[1]) and both continuous and *discontinuous* initial temperature distributions. We choose to use the RTB test case in particular for our simulations because this test case (from all test cases typically used as given in [6]) offers the largest possible gains in efficiency for an IMEX method since the stiffness of the problem derives only from the fast acoustic waves in the compressible equations.

1.2 Paper Format

Section 2 provides an overview of the test model context within which we will develop the preconditioner. In particular, we begin with the assumption that there is an existing code for applying the underlying operator (e.g. a fast parallel code, a configurable hardware implementation) and that we will therefore be using a solution algorithm that only requires the ability to apply the underlying operator (e.g. GMRES). Given these preliminary assumptions we wish to create a preconditioner while observing three basic premises:

- Premise 1: It must be possible to apply the preconditioner using only the existing matrix-free code.
- Premise 2: It must be possible to construct the preconditioner using only the existing code.

[1] In [13] it was shown that deriving the DG Schur-complement form for general boundary conditions remains an open problem.

– PREMISE 3: The system in question will be solved many times and hence the cost of constructing the preconditioner will be amortized over a great number of uses.

This section also includes an overview of the iterative solvers that we will evaluate in terms of their response to the PBNO preconditioner and several comparison preconditioners. Section 3 provides some background on polynomial preconditioning including existing generalized least squares (GLS) preconditioners to which the PBNO preconditioner will be compared. Section 4 lays out the mathematical development of the NLLS-based method for constructing the PBNO preconditioner, and explains how the preconditioner can be applied in a completely matrix-free manner. Section 4 describes how the comparision GLS preconditioners are constructed. Section 5 presents the results of the PBNO preconditioner performance in the DG model of the 2-D RTB problem in a serial computing environment. Section 6 summarizes the key results of the paper.

2 Test Model Context

The test model context for our development is a high-resolution nonhydrostatic atmospheric model (NUMA) as described in [6,7,9,8]. The combination of high-resolution and large domain size associated with global atmospheric models requires as many as $O(10^7)$ elements and $N_g = O(10^9)$ grid points (nodes). As a result, at each time-step in the IMEX time integration process there is a need to iteratively solve a very large, but sparse, linear system of the form

$$A\boldsymbol{q}^{n+1} = R(\boldsymbol{q}^n), \tag{1}$$

or, using standard generic notation

$$A\boldsymbol{x} = \boldsymbol{b}. \tag{2}$$

In Eq. (1), $\boldsymbol{q}$ is the state vector, R is a right-hand side operator, and the matrix A is square, invertible, nonsymmetric, and fixed (unless adaptive mesh refinement is used). The size of A is $4N_g \times 4N_g$ in 2-D and $5N_g \times 5N_g$ in 3-D (both are non-Schur complement forms).

At this point it is important to emphasize that system matrix A in Eq. (1) is never constructed in either a global or elemental form. Rather, the algorithms in NUMA merely accomplish the equivalent *action* of matrix A on state vector $\boldsymbol{q}$. Thus, from an implementation perspective, Eq. (2) is effectively

$$L_A(\boldsymbol{x}) = \boldsymbol{b}. \tag{3}$$

The above fact provides the motivation behind PREMISES 1 and 2 in Section 1.2 above.

To facilitate the subsequent analysis and comparison of results, it is useful to define the scaling parameter

$$\lambda_{mid} = \frac{|\lambda_{max}| + |\lambda_{min}|}{2}, \tag{4}$$

where λ_{max} and λ_{min} are the eigenvalues of system matrix A with the largest and smallest moduli, and which can be well-approximated via the Arnoldi method for a system of any size. Dividing both sides of Eq. 2 by λ_{mid} results in the equivalent system

$$(A/\lambda_{mid})\boldsymbol{x} = (\boldsymbol{b}/\lambda_{mid}), \tag{5}$$

which we will hereafter denote more concisely by

$$\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}. \tag{6}$$

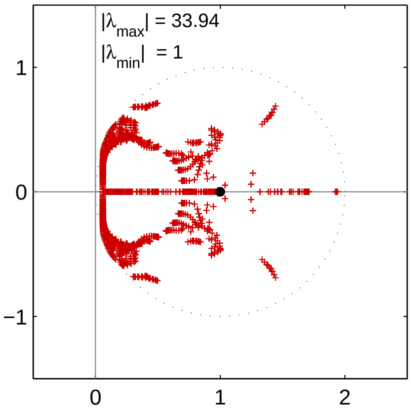


Fig. 1 Example of a complex spectrum for a matrix $\tilde{A}$ associated with a 2-D DG model of the RTB test case. The model had a coarse spatial resolution consisting of 5-by-5 elements and 5^{th} order Lagrange polynomials, based on the Legendre-Gauss-Lobatto (LGL) points, and employed a 2-stage ARK time-differencing scheme with the time-step set to produce a Courant No. of 16. The eigenvalues of the unscaled system matrix A having the largest and smallest modulus, respectively, appear in the upper left.

For a DG model of the RTB problem (for which a Schur-complement form is not presently available), the spectrum of $\tilde{A}$ is complex, confined to the right half of the complex plane, and largely contained within the unit disk as shown in Fig. 1.

Despite the need to solve Eq. (6) iteratively, the attraction of IMEX methods is that, under suitable dynamical circumstances, one may employ time-steps that are as much as 100 times (or more) greater than the maximum allowable explicit time-step. As a result, IMEX-based models can run faster than explicit models provided that the number of iterations needed to solve Eq. (6) is kept under control by a sufficiently effective preconditioner. The RTB test case is an example of a dynamical scenario that can be run at a large Courant Number in an IMEX model, and thus permits a large time-step. In [3] we described this test case and used it to show the ability of the EBSO preconditioner to accelerate the iterative solution of Eq. (6) when $\tilde{A}$ is in Schur-complement form. Herein we will use the same test case to illustrate the ability of the PBNO preconditioner to solve Eq. (6) regardless of whether $\tilde{A}$ is in Schur-complement form or not, and thus allow us to use it to precondition DG compressible flow models.

Because the matrix $\tilde{A}$ is not symmetric positive definite (SPD) our selection of suitable iterative solvers is limited to those that can handle non-SPD systems. In [3] we compared the performance of GMRES [17] based on the Arnoldi process [16, p. 165] and transpose-free BICGS [19] based on the Lanczos process [16, p. 234]. In this paper we also include the (Richardson) RICH algorithm [19, p. 22]. All three of these algorithms will be evaluated on the basis of their PBNO-preconditioned performance in a serial computing environment. The rationale in selecting these three solvers is that they demonstrate diverse combinations of computational and communication costs. For example, GMRES applies the system matrix once per iteration, but if n is the number of iterations required for convergence, then the number of dot-products performed by GMRES is $n^2/2$. By contrast, transpose-free BICGS applies the system matrix twice, but only requires $6n$ dot-products. Like GMRES, RICH applies the system matrix once per iteration. However, unlike either GMRES or BICGS, RICH can be run in a dot-product-free mode, which can be potentially advantageous in a massively-parallel computing environment.

If a linear system is to be solved only once, then the cost to construct a preconditioner prior to its use in the solution process must be kept small. However, in many IMEX applications matrix $\tilde{A}$ in Eq. (6) remains the same for hundreds or thousands of time-steps.[2] Moreover, in a typical operational numerical weather prediction model only the initial condition specification relative to the basic state changes, which changes only the right hand side of Eq. (6). As a result, matrix $\tilde{A}$ can remain unchanged for many thousands of model runs. It is a critical observation that in such situations (*i.e.*, solving the same system with different right hand sides many hundreds of thousands of times), the cost to construct the preconditioner is essentially unimportant since it will be amortized over a great number of applications. As a result, it becomes reasonable to construct the preconditioner using computationally more expensive methods. Hence, the test model context we are considering clearly satisfies PREMISE 3. In [3] we showed how a NLLS algorithm could be used to construct the EBSO preconditioner in a process that required approximately an hour of computing time on a typical desktop computer. In this paper we show that a similar NLLS algorithm can construct the PBNO preconditioner so efficiently that even when matrix $\tilde{A}$ is not in Schur-complement form the construction cost of the preconditioner is completely amortized after only a tiny fraction of the total number of time-steps in a typical integration of the RTB problem (see Section 5.4).

3 Polynomial Preconditioning Background

Generally speaking, preconditioning methods may be grouped into two classes that have been termed *implicit* and *explicit* [2], where the latter is also referred to as *sparse approximate inverse* preconditioning (See [1] for a thorough discussion of the two classes). When explicit preconditioning is employed, Eq.(6) is replaced by the equivalent system (here using left-preconditioning)

$$(K\tilde{A})\boldsymbol{x} = K\tilde{\boldsymbol{b}}, \tag{7}$$

where the matrix K is required to approximate $\tilde{A}^{-1}$ in some sense.

In polynomial preconditioning (see [12] for a thorough discussion) we require K to be a low-order polynomial in $\tilde{A}$

$$K = s(\tilde{A}) = \sum_{i=0}^{m} k_i \tilde{A}^i. \tag{8}$$

From a DG modeling perspective, an extremely attractive feature of making K a polynomial in $\tilde{A}$ is that K possesses the same degree of parallelization potential as $\tilde{A}$. Thus, incorporating a polynomial-based K into a parallel version of DG NUMA would require no additional preconditioner-specific parallelization machinery whatsoever. Polynomial preconditioning in this form clearly satisfies PREMISE 1.

Now since any polynomial in $\tilde{A}$ shares the same invariant subspaces that are common to $\tilde{A}$ and $\tilde{A}^{-1}$, the requirement that K approximate $\tilde{A}^{-1}$ is effectively the requirement that

$$\sigma(K) \approx \sigma(\tilde{A}^{-1}). \tag{9}$$

or alternatively, the requirement that

$$\sigma(K\tilde{A}) \approx \sigma(I). \tag{10}$$

[2] The reason the matrix remains constant with time is due to some judicious choices we make. First, we linearize the nonlinear system about a spatially dependent, but *time-independent* basic state. Next, in our IMEX-RK approach we only use SDIRK methods which, in the Butcher tableau, have constant diagonal terms (see [8]).

The simplest polynomial preconditioner is the Neumann preconditioner

$$s_N(\tilde{A}) = \sum_{i=0}^{m} (I - \tilde{A})^i, \tag{11}$$

which is based on the Neumann series [4] property that the right-hand side of Eq. (11) must converge to $\tilde{A}^{-1}$ as $m \to \infty$ as long as the spectrum of $\tilde{A}$ is contained within the open unit disk centered at (1,0). Neumann preconditioners contain no user-specified parameters. Thus, in situations where they can be applied successfully, they basically function as a zero-skill benchmark against which other polynomial preconditioners may be compared. However, the spectrum of the rising bubble problem shown in Fig. 1 has many small eigenvalues very close to the unit circle, and we have verified that the 2-D IMEX DG NUMA model of the RTB problem will not reliably converge at each time-step using a Neumann-type preconditioner.

A *linear* least squares approach to formally satisfying condition (10) would be the optimization problem

$$\boldsymbol{k}_{opt} \Leftarrow \min_{\lambda \in \sigma(\tilde{A})} ||1 - \lambda s(\lambda))||_2, \tag{12}$$

where vector $\boldsymbol{k}_{opt} = [k_0 \dots k_m]^T$ contains the optimal coefficients of the polynomial in Eq. (8). However, due to the intractability of (12), it has been customary (based on the maximum modulus principle) to replace (12) by

$$\boldsymbol{k}_{opt} \Leftarrow \min_{\lambda \in \mathit{\Gamma}} ||1 - \lambda s(\lambda))||_w, \tag{13}$$

where

- $\mathit{\Gamma}$ is a continuous convex contour that encloses the spectrum of a Hessenberg matrix[3] $\tilde{H}$ that arises from applying the Arnoldi process to $\tilde{A}$,
- $w(\lambda)$ is a positive weighting function,
- and the norm involved is induced by the inner product

$$(f, g) = \int_{\mathit{\Gamma}} f(\lambda) g(\lambda) w(\lambda) d|\lambda|. \tag{14}$$

Although the so-called generalized least-squares (GLS) preconditioners that result from optimization problem (13) clearly can be constructed for systems with complex spectra, often this approach is restricted to symmetric positive definite (SPD) or symmetric indefinite (SID) systems that simplify the preconditioner construction process by virtue of their real spectra [14,11,12]. Moreover, the choice of the functional form of w is typically influenced not so much by maximization of preconditioner performance, but more by the desire to obtain $\boldsymbol{k}_{opt}$ via analytical means, such as those made possible by, for instance, choosing w to be an appropriately translated/scaled Chebyshev weight (see *e.g.*, [16, p. 385]).

Past examples of GLS preconditioners for systems with a complex spectra again tend to choose w for analytical convenience (see *e.g.*, [15, p. 159] and [16, p. 387]). Furthermore, an additional issue that arises when constructing a GLS preconditioner for a system with a complex spectrum is that a degree of user-determined arbitrariness is introduced with regard to what convex form (*e.g.*, an ellipse or polygon) is to be used to enclose the system's spectrum (see *e.g.*, [15, p. 158]). Figure 2 illustrates how the hull of the spectrum of the same $\tilde{A}$ as shown in Fig. 1 is accurately represented by the spectrum of the Hessenberg matrix $\tilde{H}$ constructed from $\tilde{A}$, and also shows an example of

[3] See Section 4.1 for the details of how $\tilde{H}$ is constructed.

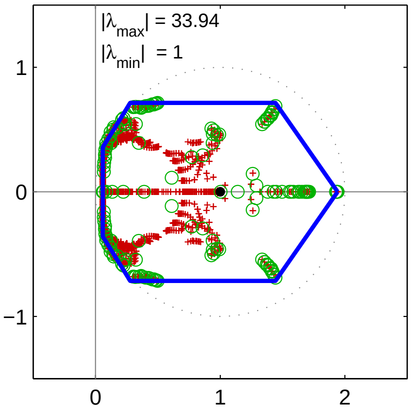


Fig. 2 As in Fig. 1, except that included is: i) the spectrum (green circles) of a 150 × 150 Hessenberg matrix $\tilde{H}$ obtained from $\tilde{A}$ via the Arnoldi process, and ii) an octagonal example of contour Γ (solid blue line) constructed so as to enclose the spectral hull of $\tilde{H}$. For comparison purposes with $\tilde{H}$, the dimension of $\tilde{A}$ is 3600 × 3600.

an octagonal contour Γ constructed from knowledge of the spectrum of $\tilde{H}$ only. We will use this octagonal Γ when we construct several GLS preconditioners (see Section 4.4) to compare with the PBNO preconditioner in terms of performance in the DG NUMA model.

4 PBNO and Comparison Preconditioner Development

4.1 PBNO Optimization Problem Formulation

In the *nonlinear* least squares approach we develop here, we begin formally by replacing optimization problem (12) with

$$\boldsymbol{k}_{opt} \Leftarrow \min_{\lambda \in \sigma(\tilde{A})} ||1 - \lambda s(\lambda))||_p, \tag{15}$$

where the subscript p is the index of a standard Euclidean p-norm. If matrix $\tilde{A}$ is $M \times M$, and we choose a power basis form for K as shown in Eq. (8), then optimization problem (15) effectively becomes

$$\boldsymbol{k}_{opt} \Leftarrow \min \left[\sum_{i=1}^{M} |1 - s(\lambda_i)\lambda_i|^p \right]^{1/p}, \tag{16}$$

Because of the poor behavior of the power basis when doing numerical computations, we will instead represent coefficients of preconditioner K using the form

$$s_L(\lambda) = \sum_{i=1}^{m+1} c_i \mathcal{L}_j(\lambda), \tag{17}$$

where

$$\mathcal{L}_j(\lambda) = \prod_{\substack{i=1 \\ i \neq j}}^{m+1} \frac{(\lambda - n_i)}{(n_j - n_i)}, \tag{18}$$

are m^{th}-order Lagrange polynomials with nodal values n_i located at the Chebyshev points appropriately translated and scaled so as to be centered in the interval $[|\lambda_{\min}|, |\lambda_{\max}|]$. By virtue of representing K via Eq. (17), optimization problem (16) is replaced by

$$\boldsymbol{c}_{opt} \Leftarrow \min \left[\sum_{i=1}^{M} |1 - s_L(\lambda_i)\lambda_i|^p \right]^{1/p}, \tag{19}$$

where vector $\boldsymbol{c}_{opt} = [c_1 \dots c_{m+1}]^T$ contains the optimal coefficients of the polynomial in Eq. (17).

In principle, optimization problem (19) provides us with a method to construct matrix K by finding the coefficients in Eq. (17), but it is likely just as intractable as optimization problem (12). Therefore, in order to advance the construction we replace the *nonlinear* least squares problem (19) with a tractable proxy via the procedure that follows.

Recall that the Arnoldi algorithm applied to an arbitrarily large M-by-M matrix $\tilde{A}$ generates the factorization

$$\tilde{A}Q = Q\tilde{H}, \tag{20}$$

where $\tilde{H}$ is an upper Hessenberg matrix of size N-by-N and N is relatively small ($N = 150$ in this paper). This upper Hessenberg matrix has a spectrum

$$\boldsymbol{\sigma}(\tilde{H}) = [\mu_1 \dots \mu_N]^T, \tag{21}$$

that can be quickly computed in its entirety using the QR algorithm [18, p. 211-224].

A key feature of $\boldsymbol{\sigma}(\tilde{H})$ is that it provides a good discrete approximation to the hull of the spectrum of the matrix $\tilde{A}$ as long as N is adequately large. This property of $\boldsymbol{\sigma}(\tilde{H})$ is illustrated in Fig. 2 for the scaled system matrix $\tilde{A}$ associated with the RTB problem.

The combination of the fact that $\boldsymbol{\sigma}(\tilde{H})$ contains the eigenvalues of $\tilde{H}$ farthest from $(1, 0)$ on the complex plane, and the fact that the residual polynomial

$$r(\lambda) = 1 - s_L(\lambda)\lambda, \tag{22}$$

contained inside optimization problem (19) tends to become large for those eigenvalues farthest from $(1, 0)$, suggests that we replace (19) with the very tractable proxy problem

$$\boldsymbol{c}_{opt} \Leftarrow \min \left[\sum_{i=1}^{N} |1 - s_L(\mu_i)\mu_i|^p \right]^{1/p}. \tag{23}$$

We note that we do not need in-depth information about the operator in order to proceed. In order to construct the proxy problem we require only the ability to apply the underlying operator and therefore this approach satisfies PREMISE 2.

In concluding this subsection we note that if $\boldsymbol{\sigma}(\tilde{H})$ is complex, then optimization problem (23) must be solved using a nonlinear least squares (NLLS) approach for all values of norm index $p > 1$. If $\boldsymbol{\sigma}(\tilde{H})$ is real, then a NLLS approach must be used for all values of $p > 2$.

4.2 PBNO Optimization Problem Solution

We can iteratively solve optimization problem (23) for any *even* value of norm index $p \geq 2$ using the Gauss-Newton (GN) algorithm as follows:

1. Create the vector-valued cost function

$$\boldsymbol{h}(\boldsymbol{c}) = \left[|1 - s_L(\mu_1)\mu_1|^{p/2}, \ldots, |1 - s_L(\mu_N)\mu_N|^{p/2}\right]^T, \tag{24}$$

   where vector $\boldsymbol{c} = [c_1 \ldots c_{m+1}]^T$ contains the coefficients c_i in Eq. (17) and the exponent form $p/2$ allows for the fact that the GN algorithm minimizes the *square* of the norm of the residual vector.
2. Approximate $\boldsymbol{h}(\boldsymbol{c})$ as a 1^{st}-order Taylor series

$$\boldsymbol{h}(\boldsymbol{c}) = \boldsymbol{h}(\boldsymbol{c_0}) + J\varDelta\boldsymbol{c}, \tag{25}$$

   where $\boldsymbol{c_0}$ is an appropriate 1^{st}-guess[4] for $\boldsymbol{c}$, and J is a finite-difference approximated Jacobian matrix given by

$$J = \left[\frac{\partial \boldsymbol{h}}{\partial c_1}, \frac{\partial \boldsymbol{h}}{\partial c_2}, \ldots, \frac{\partial \boldsymbol{h}}{\partial c_{m+1}}\right]_{\boldsymbol{c}=\boldsymbol{c}_0}. \tag{26}$$

3. Obtain the QR factorization of the Jacobian $J = QR$ and apply the minimum residual criterion $Q^T\boldsymbol{h}(\boldsymbol{c}) = \boldsymbol{0}$ to Eq. (25) to obtain the normal system

$$R\varDelta\boldsymbol{c} = -Q^T\boldsymbol{h}(\boldsymbol{c_0}), \tag{27}$$

   which is then solved for $\varDelta\boldsymbol{c}$ by back substitution.
4. If necessary, reduce $\varDelta\boldsymbol{c}$ by repeated factors of 1/2 until the descent criterion

$$\|\boldsymbol{h}(\boldsymbol{c_0} + \varDelta\boldsymbol{c})\|_2 < \|\boldsymbol{h}(\boldsymbol{c_0})\|_2 \tag{28}$$

   is satisfied.
5. Replace $\boldsymbol{c_0}$ by $\boldsymbol{c_0} + \varDelta\boldsymbol{c}$ and repeat steps 2-4 until the relative error criterion

$$\frac{\|\varDelta\boldsymbol{h}\|}{\|\boldsymbol{h}\|} < \epsilon \tag{29}$$

   is satisfied.
6. Convert the $\boldsymbol{c}_{opt}$ obtained from Steps 1-5 into the equivalent power basis coefficient vector $\boldsymbol{k}_{opt}$ so that Eq. (8) can be used when actually applying the PBNO preconditioner (via Horner's Method).

In all cases we found that letting $\epsilon = 10^{-5}$ in GN convergence criterion (29) was sufficient to ensure the GN algorithm produced a solution $\boldsymbol{c}_{opt}$ in optimization problem (23) that was of adequate precision for use in Eq. (17). The number of GN iterations required to satisfy criterion (29) depends on the order m of the PBNO preconditioner and the norm index p, varying from fewer than 10 for $m = 1$ and $p = 2$, to on the order of 50 for $m = 9$ and $p = 20$.[5]

Fig. 3(a)-(b) shows the effect of 5^{th}-order PBNO preconditioners with $p = 2$ and $p = 20$ on the spectrum of the preconditioned system matrices $K\tilde{A}$ and $K\tilde{H}$ for the RTB. The coefficients of these four PBNO preconditioners appear in Table 1 of this section. When compared to the spectra unpreconditioned system matrices $\tilde{A}$ and $\tilde{H}$ in Fig. 2, we can see that in each panel of Fig. 3 the effect of the PBNO preconditioner is to drive all the eigenvalues of $K\tilde{A}$ and $K\tilde{H}$ closer to (1,0) on the complex plane in an effort to satisfy criterion (10). Most importantly, notice in both panels of Fig. 3 that the

[4] For all cases in this paper we use $\boldsymbol{c}_0 = [1 \ldots 1]^T$, which, by virtue of the form of Eq. (17), makes $s_L(\lambda) = 1$.

[5] The range of iteration values just cited excludes the case when $p = 2$ and the spectrum of $\tilde{A}$ is positive definite, in which case optimization problem (23) is linear and GN converges in a single step.

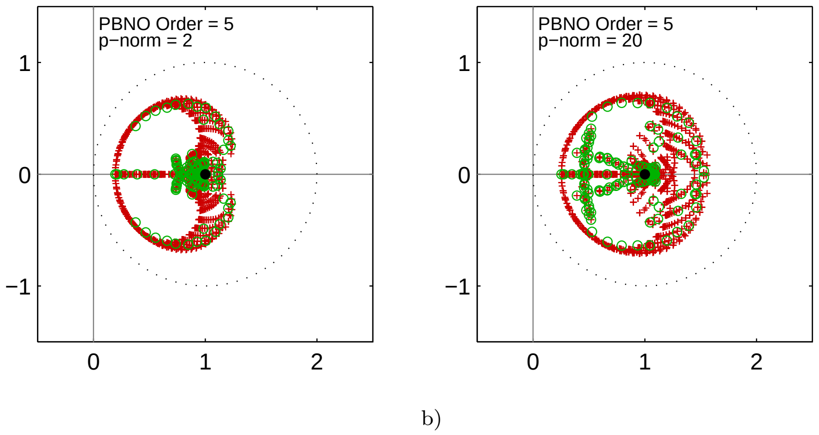


Fig. 3 (a) Spectra for the preconditioned system matrices $K\tilde{A}$ (red crosses) and KH (green circles) where $\tilde{A}$ is the same system matrix as in Fig. (2b), and where K is a 5^{th}-order PBNO preconditioner with the norm index $p = 2$. (b) As in panel (a) of this figure, except for a PBNO preconditioner using a norm index $p = 20$.

p Norm	k_0	k_1	k_2	k_3	k_4	k_5
$p = 2$	3.8319	-8.0515	10.161	-7.4160	2.8788	-0.45999
$p = 20$	5.1638	-13.367	19.799	-16.502	7.1857	-1.2669

Table 1 Coefficient values in Eq. (8) for the 5^{th}-order PBNO preconditioners that generated the spectra shown in Fig (3a-b).

spectral hull of $K\tilde{H}$ (green circles) lies virtually on top of the spectral hull of $K\tilde{A}$ (red crosses). This observation supports the assumption that tractable optimization problem (23) is a suitable proxy for the intractable problem (16).

By comparing Fig. 3(a) with 3(b) we can readily see that increasing the norm index from $p = 2$ and $p = 20$ in optimization problem (23) results in a more symmetric distribution of eigenvalues around (1,0). This is to be expected since increasing the p-norm index effectively results in a heavier weighting of the least squares residual vector components associated with the eigenvalues farthest from (1,0). We will see in Section 5 that the ability to change norm index p will allow us to create the optimal PBNO preconditioner for each of the GMRES, BICGS, and RICH iterative solvers that we will employ.

4.3 PBNO Preconditioner Construction and Application

For preconditioned system matrix spectra (red crosses) shown in Fig. 3a-b, the resolution of the RTB problem was sufficiently coarse so that we could actually construct matrix $K\tilde{A}$ and extract its *entire* spectrum via the QR method for comparison with the spectrum of the preconditioned Hessenberg matrix $K\tilde{H}$. For higher resolution test case problems, and for any operational atmospheric prediction problem, we must begin with Eq. (2) in the more numerically relevant form

$$L_A(\boldsymbol{x}) = \boldsymbol{b}, \tag{30}$$

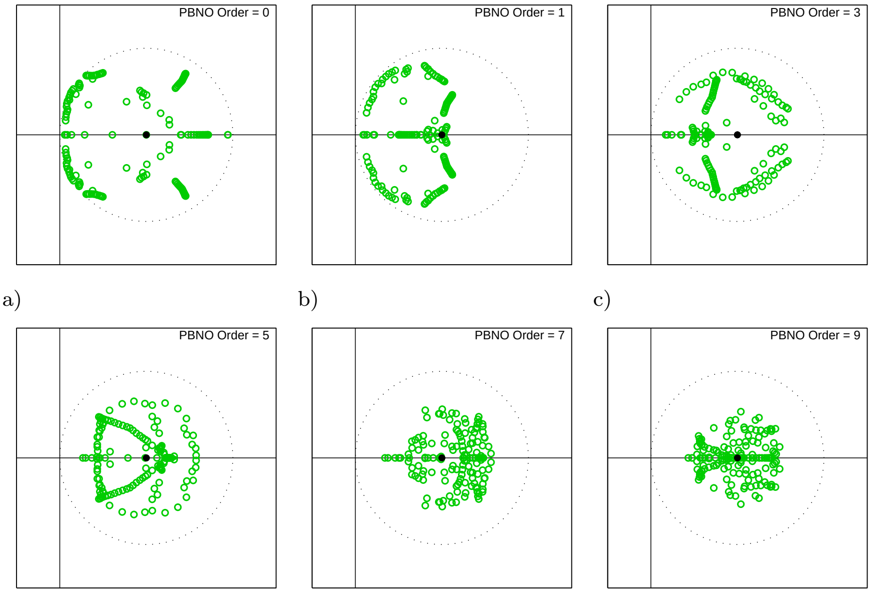
where $L_A(\ )$ is the matrix-free linear operator that accomplishes the transformational action of the unscaled system matrix A. In this case the method for constructing and applying the PBNO preconditioner is as follows:

1. Prior to the time integration phase of the numerical model, construct an *unscaled* Hessenberg matrix H via an operator-based implementation of the Arnoldi algorithm, namely:
$$L_A(Q) = QH. \tag{31}$$
2. Apply the QR method to H to obtain λ_{max} and λ_{max} in order to construct λ_{mid} in Eq. (4), which in turn enables us to create a *scaled* system matrix operator
$$L_{\tilde{A}}(\ ) = \frac{1}{\lambda_{mid}} L_A(\ ), \tag{32}$$
for use during preconditioner construction and at each step of the time integration phase of the DG model.
3. Employ the method outlined in Sections 4.1 and 4.2 to determine the coefficients of the PBNO preconditioner for the values of polynomial order m and norm index p specified by the user. We note here that Eq. (20) is replaced by the operator-based form
$$L_{\tilde{A}}(Q) = Q\tilde{H}, \tag{33}$$
and also emphasize that the *only* quantities that require storage for later use in the time-integration phase of the numerical model are the small number (≤ 10) of scalar coefficients that constitute the components of the coefficient vector $\boldsymbol{c}_{opt}$.
4. After the preconditioner is constructed via Steps 1-3 above, it is then employed at each step in the time-integration phase to solve the operator-based form of Eq. (7)
$$L_K \circ L_{\tilde{A}}(\boldsymbol{x}) = L_K(\tilde{\boldsymbol{b}}), \tag{34}$$
via the particular iterative solver specified by the user. Here it is important to emphasize that the operator L_K is implemented iteratively using Horner's Method, and thus never requires more processor memory than the modest amount[6] required by $L_{\tilde{A}}$, regardless of the polynomial order m of the PBNO preconditioner K.

We emphasize that for Steps 1-3 listed above, the greatest computational cost by far is incurred in Step 1 when the matrix H is constructed via the Arnoldi algorithm, particularly if system matrix A is large. Since we use a 150×150 Hessenberg matrix, this cost is approximately equivalent to running GMRES for 150 iterations. Similar costs would also be incurred when constructing other polynomial preconditioners (*e.g.* Chebyshev) using linear least-squares methods, as well as the cost to compute the spectrum of H in Step 2. The cost to complete Step 3, which is the only step unique to the NLLS-based PBNO preconditioner, is negligible compared to the cost of Step 1. Moreover, as we show in Section 5.4, the *total* cost of constructing the PBNO preconditioner is completely amortized very early in just a single run of the DG model provided that a realistically large Courant Number is utilized.

Fig. 4 shows the impact on the spectrum of $K\tilde{H}$ due to a set of PBNO preconditioners of increasing order constructed for a DG model RTB problem with five times the spatial resolution of the analogous coarse-resolution model having the spectra shown in Fig. 3. Notice the similarity of the spectra of the unpreconditioned scaled Hessenberg matrix $\tilde{H}$ in Fig. 4(a) with the analogous matrix $\tilde{H}$ for the course-resolution model shown in Fig. 2. Likewise, note the similarity of the spectra of the matrices $K\tilde{H}$ (green circles) in Fig. 3(b) and Fig. 4(d), where, in each case a 5^{th}-order PBNO preconditioner with $p = 20$ was employed. Finally, notice in Fig. 4(f) the high degree of axisymmetry about (1,0) exhibited by the spectrum of $\tilde{H}$ when a 9^{th}-order PBNO preconditioner is employed.

[6] Recall that in NUMA neither the unscaled system matrix A nor the scaled system matrix $\tilde{A}$ is ever constructed.



a) b) c)

d) e) f)

Fig. 4 From top left to right (a)-(c) and bottom left to right (d)-(f). Spectrum of the scaled 150 × 150 Hessenberg matrix $\tilde{H}$ derived from a DG model of the RTB problem using 25 × 25 elements, 5^{th}-order polynomials (based on LGL points), 2nd-order-accurate ARK time differencing, and a Courant No. of 16. The corresponding scaled system matrix $\tilde{A}$ has dimensions of $22,550 \times 22,500$.

4.4 Comparison GLS Preconditioners

Here we outline the method set forth by van Gijzen [5], which constructs a GLS preconditioner by *numerical* means, and has the advantage of allowing us to choose any positive weighting function $w(\lambda)$ we desire. The first step is to view the polynomial portion of the integrand in optimization problem (13) as the function

$$f(\boldsymbol{k}, \lambda) = 1 - \lambda s(\lambda), \tag{35}$$

and then to create the inner product-based cost function

$$F(\boldsymbol{k}) = \int_{\Gamma} f(\boldsymbol{k}, \lambda)\overline{f(\boldsymbol{k}, \lambda)} w(\lambda) d|\lambda|. \tag{36}$$

For an m^{th}-order GLS preconditioner, the $\boldsymbol{k}_{opt}$ that minimizes the cost function is the solution to the $(m+1)\times(m+1)$ linear system arising from the stationary-point conditions

$$\frac{\partial F}{\partial k_i} = 0 \qquad i \in [0, m]. \tag{37}$$

The reader is referred to van Gijzen [5, p. 96-97] for the system-matrix entry and right-hand side vector component formulas resulting from Eqs. (37) when preconditioner K is represented via a power basis as in Eq. (8). Although the use of a power basis has the potential for numerical instability as polynomial order m becomes large, we have verified van Gijzen's claim that no such problem occurs for lower order polynomials. Specifically, when $\Gamma = [0, 4]$ and a Chebyshev $w(\lambda)$ is used, our numerical computations for $m \leq 8$ approximate the exact results presented by Saad [16, p. 385] out to 8 significant digits.

Using the above approach we have constructed GLS preconditioners using both uniform weighting (hereafter GLSU) and generalized Chebyshev weightings (hereafter

GLS Variant	k_0	k_1	k_2	k_3	k_4	k_5
GLSU	3.5599	-7.0695	8.4318	-5.9007	2.2272	-0.3494
GLSC	3.4839	-7.2185	9.0164	-6.4720	2.4619	-0.3848

Table 2 Coefficient values in Eq. (8) for the 5^{th}-order GLSU and GLSC preconditioners created using the octagonal contour Γ shown in Fig. 2.

GLSC) applied to all sides of the octagonal contour Γ shown in Fig. 2. The coefficients of resulting $\boldsymbol{k}_{opt}$ in each case are shown in Table 2 for comparison with the analogous PBNO preconditioner coefficients shown in Table 1.

5 Results in a Serial Computing Environment

In order to adequately map out the large parameter space associated with:

- the DG variant of NUMA2D[7]
- a range of possible time-steps with Courant No. as high as 32,
- three iterative solvers (GMRES, BICGS, RICH),
- preconditioner order as high as 9^{th}-order,
- and the adjustable p-norm index in PBNO optimization problem (23),

we initially use a coarse spatial resolution, a low order time integration scheme, and a relatively short simulation time of 100s. The model domain is spatially discretized using a 25-by-25 element grid and 5^{th}-order Lagrange polynomials (based on LGL points) within each element, and a 2^{nd}-order accurate Additive Runge-Kutta (ARK2) time integration scheme [8]. A set of six different time-steps is employed and corresponds to a Courant No. as small as 2 and as large as 32.[8] As the analysis proceeds we will include selected results that employ 9^{th}-order spatial resolution, higher order ARK methods, and a model simulation time of 700s to show that conclusions based on the coarse simulation runs are justified.

5.1 Solver Performance Dependence on PBNO p-Norm Index

Recall that in optimization problem (23) the index of the p-norm employed can be varied. The effect of changing index p on the performance of the GMRES, BICGS, and RICH iterative solvers using PBNO preconditioners of up to 9^{th}-order is illustrated in Fig. 5 for DG-NUMA2D. Increasing the value of index p from 2 to 10 to 20 moderately improves the performance of GMRES (Fig. 5(a)-(b)) and BICGS (Fig. 5(c)-(d)), and significantly improves the performance of RICH (Fig. 5(e)-(f)). Based on the results shown, a setting of $p = 10$ would be appropriate when using either the GMRES or BICGS solver, and a setting of $p = 20$ would be an appropriate choice when using the RICH solver. In comparing Figs. 5(b) and 5(d), it is important to note that the wall-clock time of the RICH solver (with $p = 20$) is of the same order as both GMRES and BICGS regardless of PBNO preconditioner order m. A summary of the p-norm index values used in all subsequent model runs are provided in Table 3.

[7] NUMA2D refers to the two-dimensional version of the NUMA model.

[8] A Courant No. of 32 is the largest for which the RTB problem will run to completion (*i.e.,* bubble at top of domain at 700s) without exceeding the CFL limit of the explicit part of the IMEX method.

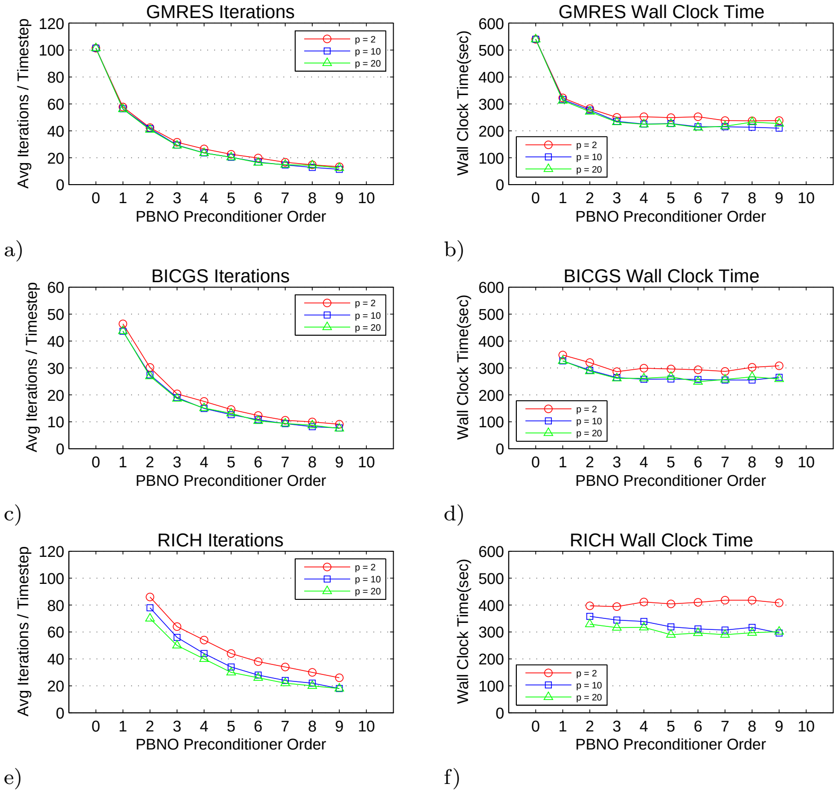


Fig. 5 (a)-(b) Effect of PBNO preconditioner p-norm index on GMRES Iterations/Time-step and Wall Clock Time, respectively, when employing a time-step corresponding to a Courant No. of 16. (c)-(d) As in (a)-(b), except for the BICGS iterative solver. The missing data points for PBNO order $m = 0$ reflects the fact that unpreconditioned BICGS solver would not reliably converge at every time-step. (e)-(f) As in (a)-(b), except for the RICH iterative solver. The missing data points reflect the fact that the RICH solver would not converge for PBNO order $m < 2$.

5.2 Comparison of PBNO and GLS Preconditioner Performance

In Figure 7 we provide comparisons of how effective 3^{rd}-order and 7^{th}-order PBNO and GLS preconditioners are in improving the convergence rate of the GMRES (Fig. 7(a)-(b)) and RICH (Fig. 7(c)-(d)) iterative solvers for one particular time-step in the RTB model. It is important to emphasize that because the system matrix $\tilde{H}$ remains *unchanged* throughout the entire model run, the "snap-shots" provided in Fig. 7 are completely representative of the relative performance of the preconditioners at all time-steps during the time integration process. The performance comparison using the GMRES solver (Fig. 7(a)-(b)) clearly reveals that for both 3^{rd}-order and 7^{th}-order polynomials:

- The GLSC preconditioner makes essentially negligible improvement over the GLSU preconditioner, which indicates that, at least for the RTB problem, the mathematical advantages of choosing a Chebyshev weighing function do not translate into any meaningful performance improvement over a uniform weighting function.
- The PBNO preconditioner requires approximately 20% fewer iterations of GMRES to achieve a relative residual of 10^{-6} compared to GLSU.

The relative performance of the PBNO, GLSU, and GLSC preconditioners using the BICGS solver were found to be essentially the same as for GMRES and are not shown.

	GMRES	BIGCS	RICH
Norm index	$p = 10$	$p = 10$	$p = 20$

Table 3 Norm index values used in optimization problem (23) when constructing PBNO preconditioners for the three different iterative solvers.

The performance comparison using the RICH solver (Fig. 7(c)-(d)) clearly reveals:

- The RICH solver will not converge without preconditioning
- The GLSC preconditioner actual performs slightly worse than the GLSU preconditioner. This is because although the GLSC produces a tighter spectral grouping, it fails to center the grouping within the unit circle about (1,0), which is essential for accelerating the RICH solver.
- The PBNO preconditioner requires approximately 30% fewer iterations of RICH to achieve a relative residual of 10^{-6} compared to GLSU. This increase in RICH solver convergence rate illustrates the fact that *by design* the PBNO preconditioner attempts to center the spectrum of the preconditioned system matrix $K\tilde{A}$ within the unit circle about (1,0) (recall Fig. 4e for a 7^{th}-order PBNO preconditioner).

Since the foregoing results clearly indicate that the PBNO preconditioner will significantly outperform the comparison GLSU and GLSC preconditioners at all time-steps, no further reference to the GLS preconditioners will be made, and all the results that follow focus on analyzing the properties of the PBNO preconditioner with regard to the choice of iterative solver, time integrator order, model initial conditions, etc.

5.3 Relative Performance of PBNO-preconditioned Solvers

The effect of the PBNO preconditioners on the iterations/time-step and wall clock time of GMRES, BICGS, and RICH solvers as a function of RTB problem Courant No. are shown in Fig. 7. Comparing Figs. 7(b),(d) and (f) reveals that in terms of minimum wall clock time all three solvers run more efficiently when higher order preconditioning is combined with large Courant No. In particular:

- when employing GMRES the model runs the fastest when a 9^{th}-order preconditioner is combined with the maximum Courant No. of 32.
- when employing BICGS the model runs the fastest when a 7^{th}-order preconditioner is combined with the maximum Courant No. of 32.
- when employing RICH the model runs the fastest when a 9^{th}-order preconditioner is combined with the maximum Courant No. of 32.

When looking over Figs. 7(b), (d) and (f) it is clear that the lowest wall clock time tends to occur when using the largest Courant No. of 32 (and thus longest time-step). However, the preconditioner order needed to achieve the lowest wall clock time when running the model at Courant No. 32 varies with the iterative solver used. Figure 8 provides an example of the PBNO-preconditioned model performance when running at the maximum Courant No. of 32 and employing an ARK4 [10] time integrator. With regard to iterations per time-step (Fig. 8(a)), important points to note are that:

- The performance of the RICH solver is only slightly worse than the GMRES and BICGS solvers for low PBNO order, but nearly matches the performance of GMRES for PBNO order greater than 4.

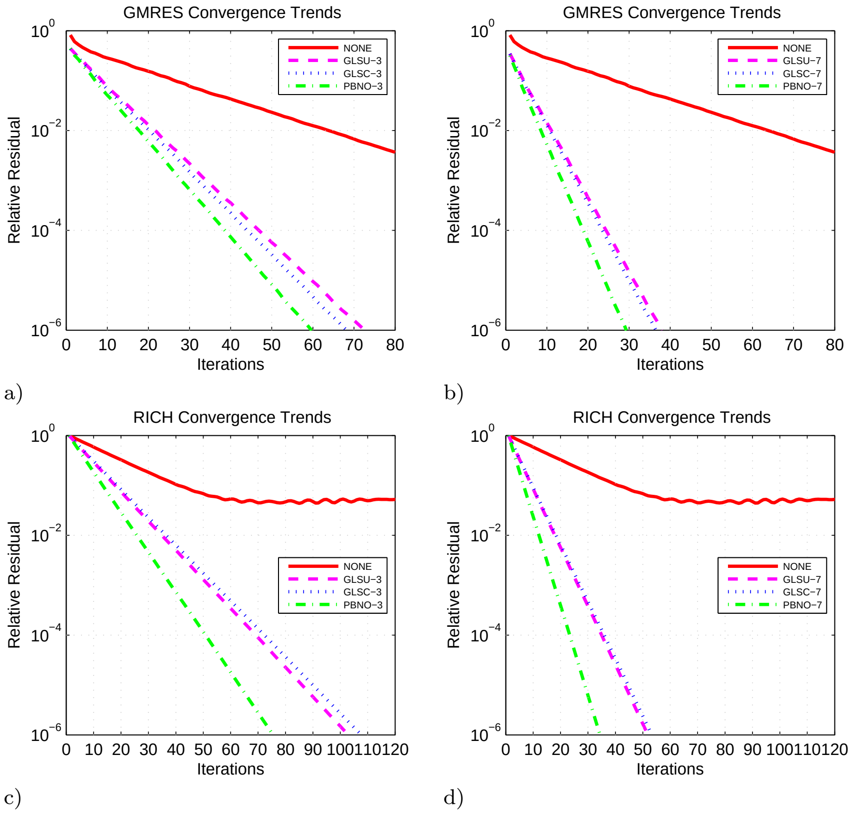


Fig. 6 (a) A representative example of the relative residual convergence rates exhibited by the GMRES iterative solver *at each time-step* of the RTB problem using the 3^{rd}-order preconditioners shown in the legend and the following model specifications: 25-by-25 element grid and 5^{th}-order Lagrange polynomials (based on LGL points) within each element, an ARK2 time integrator and a Courant No. of 16. (b) As in (a), except for 7^{th}-order preconditioners. (c)-(d) As in (a) and (b) respectively, except for using the RICH iterative solver.

- Although BICGS has significantly fewer iterations per time-step than GMRES, it must be remembered that this advantage tends to be offset, to some degree, by the fact that BICGS applies the preconditioned system matrix twice at each time-step.

With regard to wall clock time (Fig. 8(b)), important points to note are that:

- The 1^{st}-order preconditioning of GMRES or BICGS results in a dramatic reduction in wall clock time compared to unpreconditioned GMRES (the only solver for which DG-NUMA2D would run without preconditioning).
- The performance of all three solvers is very similar for PBNO order $m \geq 2$ and we can see that a modest reduction in wall clock time occurs as PBNO order increases.
- The competitiveness of the RICH algorithm for PBNO order $m \geq 2$ is particularly noteworthy given that the algorithm is run in a dot-product-free mode in NUMA2D.

5.4 PBNO Preconditioner Construction Cost Amortization

Recall that Premise 3 formalizes the assumption that the same preconditioner can be used many times, and that one might expect that this will amortize the cost of con-

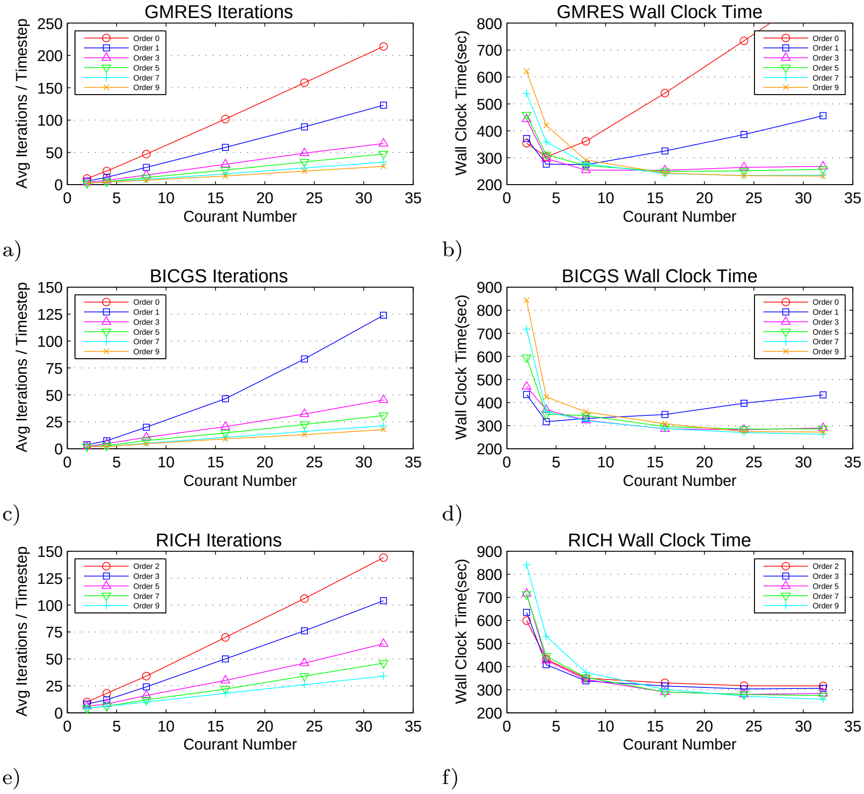


Fig. 7 (a)-(b) Iterations/time-step (a) and wall clock time (b) as a function of Courant No. exhibited by DG-NUMA2D preconditioned with the orders of the PBNO-preconditioner shown in the legend. For these results DG-NUMA2D employed 5^{th}-order Lagrange polynomials (based on LGL points), an ARK2 time integrator, and RTB simulation time of 100s. (c)-(d) As in (a) and (b), except for using the BICGS solver. The missing data points for 0-order preconditioning indicates that the model would not run to completion when employing the unpreconditioned BICGS solver. (e)-(f) As in (a) and (b), except for using the RICH solver.

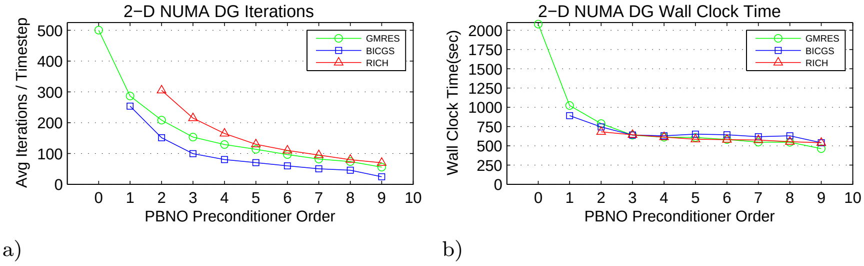


Fig. 8 Comparison of iterations/time-step (a) and wall clock time (b) as a function of PBNO preconditioner order for DG-NUMA2D. Missing data points indicate that the model would not run to completion using that particular combination of iterative solver and PBNO order. The model employed 5^{th}-order Lagrange polynomials (based on LGL points), a Courant No. of 32, an ARK4 time integrator, and RTB simulation time of 100s.

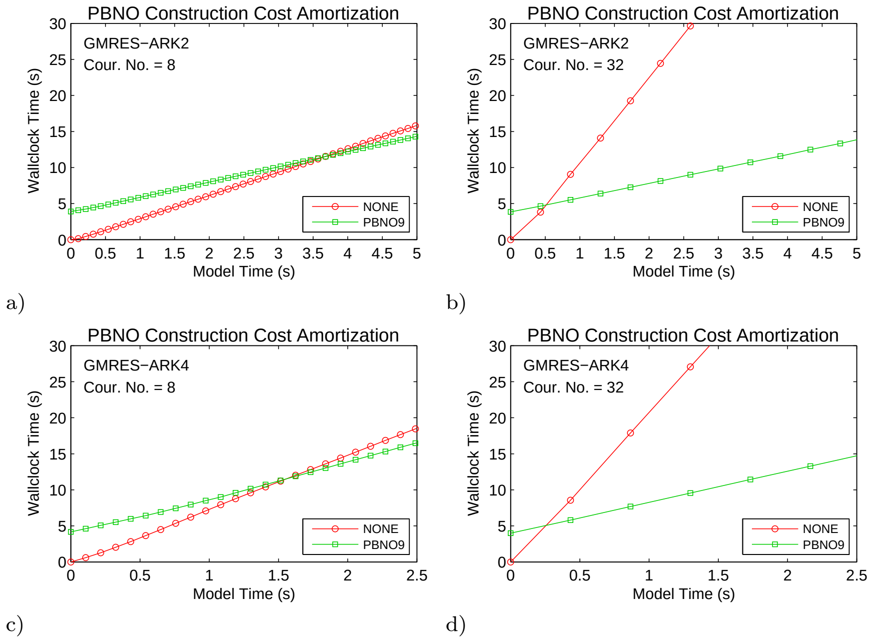


Fig. 9 (a)-(d) Graph of wall-clock time versus model simulation time when running the RTB problem using unpreconditioned GMRES (red-circles) and GMRES using a 9^{th}-order PBNO preconditioner (green squares) for the various combinations of Courant No. and ARK time integrator accuracy shown in the upper left of each panel. The horizontal distance between adjacent circles or squares indicates the length of the time-step employed. In all the panels the model domain is spatially discretized using a 25-by-25 element grid and 5^{th}-order Lagrange polynomials (based on LGL points) within each element.

structing the preconditioner over a great number of repeated applications. The algebraic justification for this premise is that, within a suitably formulated IMEX modeling context, the linear operator represented by matrix $\tilde{A}$ in Eq. (7) does not vary from time-step to time-step, and thus matrix K (*i.e.*, the PBNO preconditioner) does not need to change. Verification of the validity of Premise 3 is provided in Figs. 9(a)-(d) which plots wall-clock time as a function of model simulation time for runs of NUMA2D-DG that employ both unpreconditioned GMRES (red circles) and GMRES preconditioned with a 9^{th}-order[9] PBNO preconditioner (green squares).

First, notice that all the graphs are nearly straight lines, which reflects the fact that matrix $\tilde{A}$ in Eq. (7) does not change. The slight departures from linearity are a manifestation of the fact that the average number of GMRES iterations per time-step has a slight dependence on vector $\tilde{\boldsymbol{b}}$ in Eq. (7), which changes from time-step to time-step due to its dependence on the evolving RTB state vector (recall Eqs. (1) and (2)).

Notice that in each panel of Fig. 9 the preconditioned GMRES graph starts with a wall-clock time of approximately 4 seconds, which reflects the upfront cost of constructing the PBNO preconditioner before RTB simulation begins. However, the key observation is that in each panel the preconditioned GMRES wall-clock time graph crosses the unpreconditioned GMRES wall-clock time graph within the first few seconds of model simulation time. At the cross-over point the cost of constructing the PBNO preconditioner has been completely amortized, which means that after the cross-over point the preconditioned model is requiring less wall-clock time than the unprecondi-

[9] By making the polynomial order the largest we consider in this paper (*i.e.*, 9^{th}-order) we maximize the number of Gauss-Newton iterations required during the construction of the preconditioner, and thus maximize the construction cost.

tioned model. For the 2^{nd}-order-accurate ARK time integrator, when we use a time-step corresponding to a Courant No. of 8, the cross-over point occurs after 34 time-steps (Fig. 9(a)); when we increase the Courant No. to 32 the cross-over point drops dramatically to just after the 1st time-step (Fig. 9(b)). For the 4^{th}-order-accurate ARK time integrator, when using a time-step corresponding to a Courant No. of 8 the cross-over point occurs after only 14 time-steps (Fig. 9(c)); increasing the Courant No. to 32 brings the cross-over point to before the completion of the 1st time-step (Fig. 9(d)). The increasing rapidity with which the construction cost of the PBNO preconditioner is amortized as ARK order and Courant No. are increased reflects the fact that the construction cost of the preconditioner is roughly equivalent to 150 GMRES iterations (as determined by the size of $\tilde{H}$) *regardless* of the Courant No. or ARK order used. By contrast, the average number of GMRES iterations per time-step increases as either the order of the ARK method or the size of the Courant No. are increased.

The results in this subsection clearly show that Premise 3 is easily satisfied even if the PBNO preconditioner is applied to only one run of NUMA2D-DG. In the next subsection, we show that, once constructed, the same PBNO preconditioner can be used for multiple runs of the model with different initial conditions, thus making the cost of constructing the preconditioner even more inconsequential.

5.5 Long-Duration, High-Resolution Simulation Comparisons

As mentioned earlier, to expeditiously map out the large parameter space associated with multiple PBNO variants, iterative solvers, *etc*, we initially presented results based on coarse spatial resolution, a low-order time integration scheme, and relatively short simulation time of 100s. To ensure that PBNO preconditioning up to order $m = 9$ produces stable and consistent results for longer RTB simulation times, regardless of the iterative solver used, we ran the problem to 700s, by which time the bubble has already impacted the top of the domain. For these runs we employ a problem domain that is spatially discretized using a 25-by-25 element grid and 9^{th}-order Lagrange polynomials (based on LGL points) within each element, and we use a 4^{th}-order accurate Additive Runge-Kutta (ARK4) time integration scheme.[10]

For these runs we also used two different initial conditions:

- the continuous cosine-based profile shown in Fig. 10(a) on which all runs shown earlier in this paper are based,
- and a discontinuous profile shown Fig. 10(b) to show that PBNO preconditioned NUMA2D-DG can effectively handle strong state variable gradients, as well as show that changing the initial condition does not require recomputation of the PBNO preconditioner coefficients.

For the purpose of comparison, Figures 10(c) and (d) show the RTB potential temperature fields for the continuous and discontinuous cases, respectively, at 100 seconds of model simulation time, which is well before significant deformation of the bubble occurs in either simulation. In the discontinuous case (Fig. 10(d)), one can see that the inclusion of an artificial kinematic viscosity of 1.0 m^2/s has smoothed the initial discontinuity into a strong gradient.[11] A smaller artificial kinematic viscosity of $0.01 m^2/s$ is

[10] The total number of grid points in this simulation is 50,625 with each grid point having 4 variables for a total of 202,500 degrees of freedom. Using a time-step of dt=0.148287 means that the linear system has to be solved $\frac{700 \cdot k}{dt} = 18,882$ times during the 700s simulation, where k=4 denotes the number of implicit Runge-Kutta stages used in the ARK method.

[11] We emphasize here that the use of artificial viscosity is not necessary for NUMA2D-DG to run the discontinuous RTB case to completion, and has no effect on the performance of the PBNO precondi-

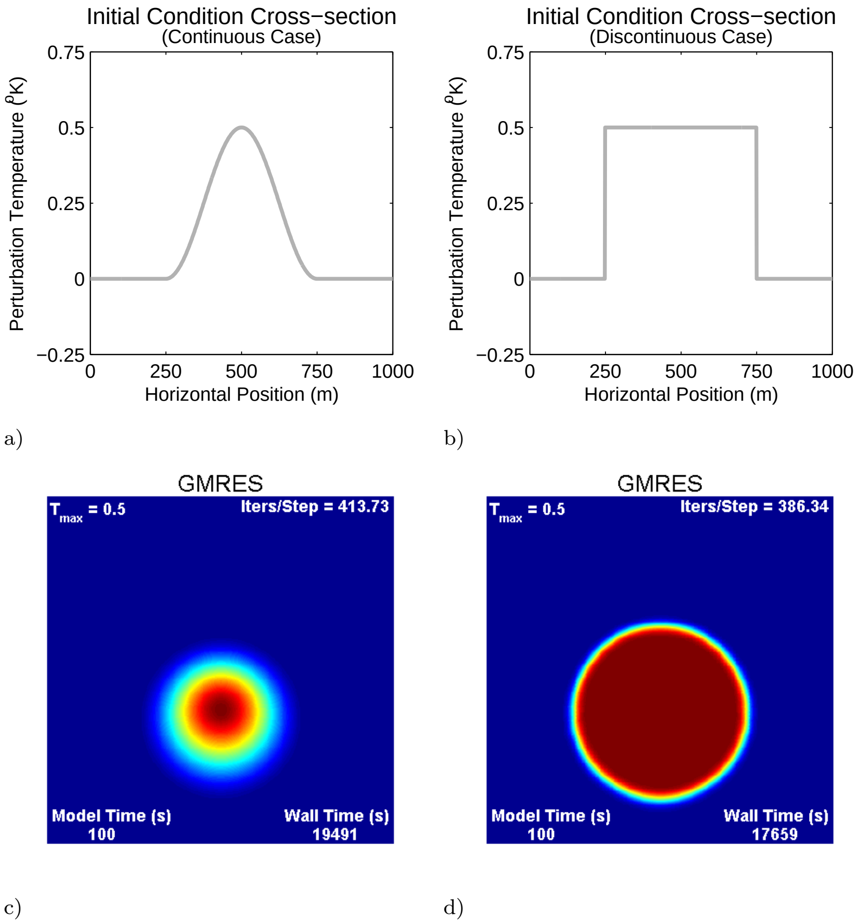
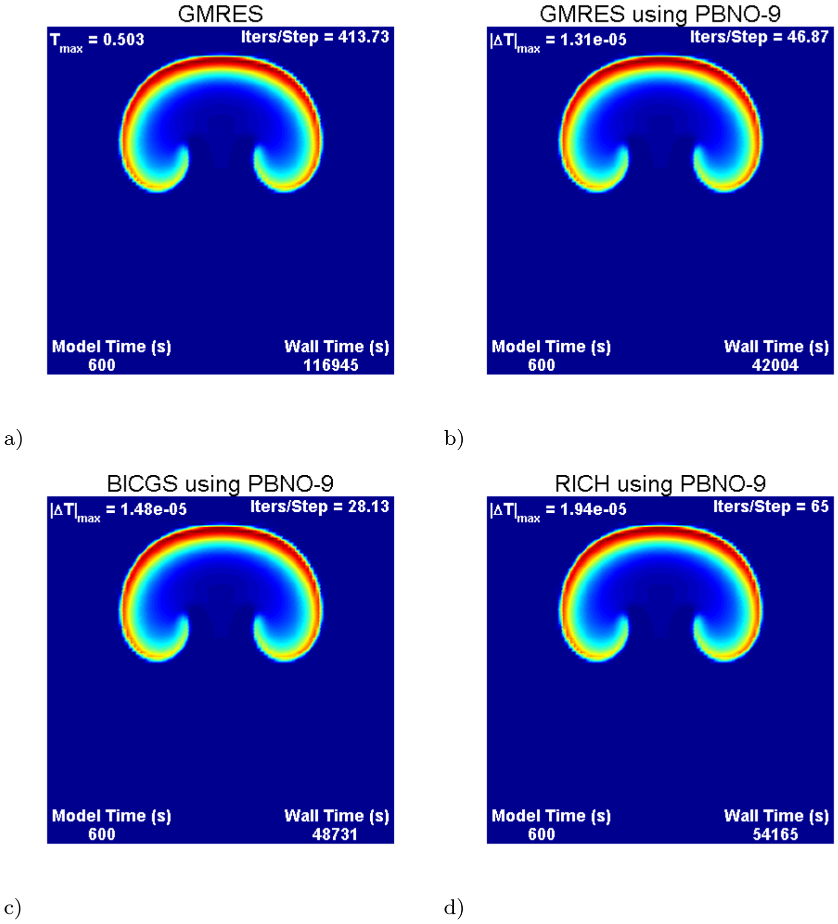


Fig. 10 (a)-(b) RTB cross-sections for the continuous cosine-based and discontinuous step-function-based initial conditions respectively. (c)-(d) Potential temperature fields at 100s corresponding to the initial conditions shown in panels (a) and (b), respectively.

also used in the continuous case, but only for the purpose of making the cost of running NUMA2D-DG comparable for the two initial conditions.

Potential temperature fields that result from running the model using the continuous initial condition (Fig. 10(a)) and employing unpreconditioned GMRES, 9^{th}-order PBNO-preconditioned GMRES, BICGS, and RICH are depicted in Fig. 11(a)-(d), respectively. The fields shown are at the 600s point in the 700s simulation, which is approximately the time when the bubble attains its maximum upward speed. It is visually evident that all four runs closely reproduce the fine structure of the rising bubble. The small values of $|\Delta T|_{max}$ shown in Fig. 11(b)-(d) provide numerical confirmation that 9^{th}-order preconditioning of the GMRES, BICGS, and RICH solvers is not causing any significant dispersion or phase lag relative to the unpreconditioned GMRES field (Fig. 11(a)). A comparison of wall clock time values in the lower right of each panel shows that relative to unpreconditioned GMRES (which is the only solver that runs without preconditioning in DG NUMA2D):

tioner. Rather the numerical viscosity has been added solely for the purpose of avoiding the generation of spurious eddies associated with imposing a curving discontinuous initial condition on a rectangularly discretized model domain, and thus allowing the continuous and discontinuous runs to produce visually comparable results.



a) b)

c) d)

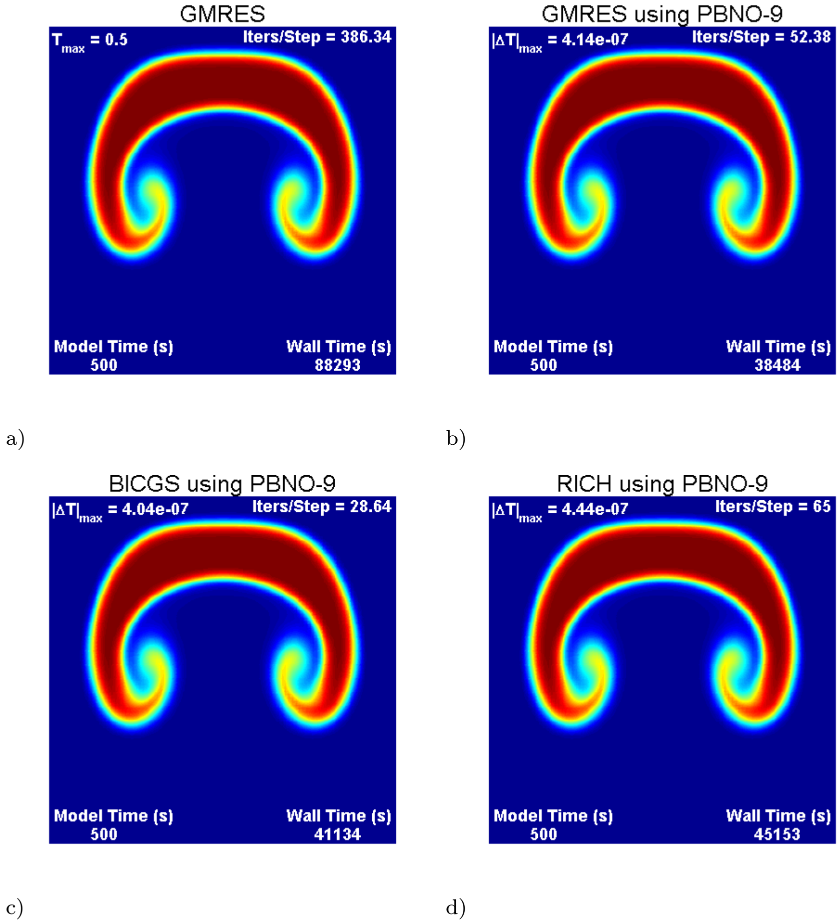
Fig. 11 (a)-(d) Potential temperature fields for the *continuous* RTB test case at 600s in a 700s simulation time run of the model. Model setup specifications are as described in the text. The heading on each panel identifies the iterative solver and order of PBNO preconditioning employed. Iterations per time-step and wall clock time in seconds appear in the upper and lower right of each panel, respectively. The value in the upper left of panel (a) is the maximum potential temperature in the bubble. The value in the upper left of panels (b)-(c) is the infinity norm of the difference between the unpreconditioned GMRES potential temperature state vector in (a) and the respective potential temperature state vector in panels (b), (c), and (d).

– preconditioned GMRES is running ≈ 2.78 times faster,
– preconditioned BICGS is running ≈ 2.40 times faster,
– and preconditioned dot-product-free RICH is running ≈ 2.16 times faster.

A comparison of average iterations per time-step values in the upper right of each panel of Fig. 11 shows that relative to unpreconditioned GMRES:

– preconditioned GMRES uses ≈ 8.83 times fewer iterations,
– preconditioned BICGS uses ≈ 14.7 times fewer iterations,
– and preconditioned RICH uses ≈ 6.37 times fewer iterations.

Potential temperature fields that result from running the model using the discontinuous initial condition (Fig. 10(b)) and employing unpreconditioned GMRES, 9^{th}-order



a) b)

c) d)

Fig. 12 (a)-(d) As in Figs. 10(a)-(d), except for the *discontinuous* RTB test case at 500s.

PBNO-preconditioned GMRES, BICGS, and RICH are depicted in Fig. 12(a)-(d), respectively. The fields shown are at the 500s point in the 700s simulation, which is approximately the time when the more energetic discontinuous bubble reaches the same height and state of development as the less energetic continuous bubble. As in the continuous RTB case, it is again visually evident that all four runs closely reproduce the fine structure of the rising bubble, as well as maintaining the strong gradient of temperature seen at 100 s in Fig. 10(d). The small values of $|\Delta T|_{max}$ shown in Fig. 12(b)-(d) provide numerical confirmation that 9^{th}-order preconditioning of the GMRES, BICGS, and RICH solvers is not causing any significant dispersion or phase lag relative to the unpreconditioned GMRES field (Fig. 12(a)).

As we conclude this section, we again wish to emphasize that the same PBNO preconditioner that was constructed for the continuous bubble case was also utilized in the discontinuous bubble case, and could have been used with an unlimited number of different initial conditions without incurring any additional construction cost.

6 Conclusion

We have introduced a method for constructing a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm and have shown that this polynomial-based

NLLS-optimized (PBNO) preconditioner significantly outperforms two generalized linear least squares (GLS) preconditioners when running a 2-D discontinuous Galerkin (DG) compressible flow model of a rising bubble (RTB) test case using implicit-explicit (IMEX) time integrators. The DG model with PBNO preconditioner achieves significant reduction in GMRES iteration counts and model wall-clock time. Comparisons of the ability of the PBNO preconditioner to improve model performance when employing BICGS and RICH have also been included. In particular, we have shown that higher order PBNO preconditioning of RICH (which is run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment. In addition, since the method used to construct the PBNO preconditioner can, without modification, handle both positive definite and complex spectra, we believe that the PBNO preconditioning approach is, for certain types of problems, an attractive alternative to existing GLS polynomial preconditioners based on linear least-squares methods.

Acknowledgements The authors gratefully acknowledge the support of the Computational Mathematics program of the Air Force Office of Scientific Research, the Office of Naval Research through program element PE-0602435N, and the National Science Foundation (Division of Mathematical Sciences) through program element 121670. We also would like to thank Michal Kopera and several anonymous reviewers for their helpful suggestions for improving the manuscript.

Conflict of Interest: The authors declare that they have no conflict of interest.

References

1. M. Benzi, *Preconditioning Techniques for Large Linear Systems: A Survey*, J. Comput. Phys., 182 (2002), 418-477.
2. M. Benzi and M. Tuma, *A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 19 (1998) pp. 968-994.
3. L.E. Carr III, C.F. Borges and F.X. Giraldo, *An Element-Based Spectrally-Optimized Approximate Inverse Preconditioner for the Euler Equations*, SIAM J. Sci. Comput., 34 (2012) pp. B392-420.
4. P.F. Dubois, A. Greenbaum, and G.H.Rodrigue, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors, Computing, 22 (1979), pp. 257-268.*
5. M.B. van Gijzen, *A polynomial preconditioner for the GMRES algorithm*, J. Comp. Appl. Math., 59 (1995), pp. 9197.
6. F.X. Giraldo and M. Restelli, *A Study of Spectral Element and Discontinuous Galerkin Methods for the Navier-Stokes Equations in Nonhydrostatic Mesoscale Atmospheric Modeling: Equation Sets and Test Cases*, J. Comp. Phys., 227 (2008), pp. 3849-3877.
7. F.X. Giraldo, M. Restelli, and M. Lauter, *Semi-Implicit Formulations of the Navier-Stokes Equations: Applications to Nonhydrostatic Atmospheric Modeling*, SIAM J. Sci. Comput., 32 (2010), pp. 3394-3425.
8. F.X. Giraldo, J.F. Kelly, and E.M. Constantinescu, *Implicit-Explicit formulations for a 3D nonhydrostatic unified model of the atmospheric (NUMA)*, SIAM J. Sci. Comput., 35 (2013), pp. B1162-1194.
9. J. F. Kelly and F.X. Giraldo, *Continuous and Discontinuous Galerkin Methods for a Scalable 3D Nonhydrostatic Atmospheric Model: Limited Area Mode*, J. Comp. Phys., Vol. 231, pp.7988-8008.
10. C. Kennedy and M. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44 (2003), pp. 139181.
11. Y. Liang, *Generalized Least-Squares Polynomial Preconditioners for Symmetric Indefinite Linear Equations*, Parallel Comput., 28 (2002), 323-341.
12. Y. Liang, *The Use of Parallel Polynomial Preconditioners in the Solution of Systems of Linear Equations*, Ph.D. dissertation, University of Ulster, 2005.
13. M. Restelli and F.X. Giraldo, *A Conservative Semi-Implicit Discontinuous Galerkin Method for the Navier-Stokes Equations in Nonhydrostatic Mesoscale Atmospheric Modeling*, SIAM J. Sci. Comp., Vol. 31, 2231-2257 (2009).
14. Y. Saad, *Iterative Solution of Indefinite Symmetric Linear Systems by Methods Using Orthogonal Polynomials Over Two Disjoint Intervals*, SIAM J. Numer. Anal., 20 (1983), 784-811.
15. Y. Saad, *Least Squares Polynomials in the Complex Plane and Their Use for Solving Nonsymmetric Linear Systems*, SIAM J. Numer. Anal., 24 (1987), 155-169.
16. Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia PA, 2003.
17. Y. Saad and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.

18. L.N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia PA, 1997.
19. H.A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631-644.
20. H.A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, New York NY, 2003.